Modelo CORMAS Agualoca

C. S. Rabak Guilherme M. Dias Lucie Clavel Raphaèle Ducrot

São Paulo, 8 de outubro de 2006.

Abstract

This is a work in progress of the documenting of Agualoca modeling work.

Resumo

Este é um documento em preparação para documentar a modelagem do Agualoca.

1 Introdução

Agualoca é um simulador de uma bacia hidrográfica hipotética para uso em um jogo de papéis (*Role Playing Game* — RPG) de mesmo nome. No apêndice C temos a representação esquemática dessa bacia hipotética com a notação introduzida por [Duc04].

Este documento descreve a implementação da parte do simulador hidráulico do Agualoca. Essa implementação foi feita usando-se a linguagem Smalltalk na versão VisualWorks da Cincom Systems sobre uma plataforma (também programada nessa linguagem) multiagentes denominada CORMAS descrita na referência [BBPP98] e nos tutoriais [CIR03] e [CIR02].

Um simulador é sistema computacional que busca aproximar da melhor maneira possível as características dos sistema físico representado-o através de descrições matemáticas. O computador automatiza os cálculos necessários e simplifica a apresentação dos resultados de forma que sejam entendidas pelos interessados no problema em estudo.

Como menciona [Rob02] os modelos de simulação são de fácil entendimento mesmo por não especialistas e podem ser introduzidos até mesmo a leigos na matéria, de forma similar na discussão do projeto Negowat apresentada em [DPPT04] indica-se uma abordagem similar.

2 Agualoca

O modelo do Agualoca é baseado no Spatmas, outro simulador desenvolvido no projeto Negowat.

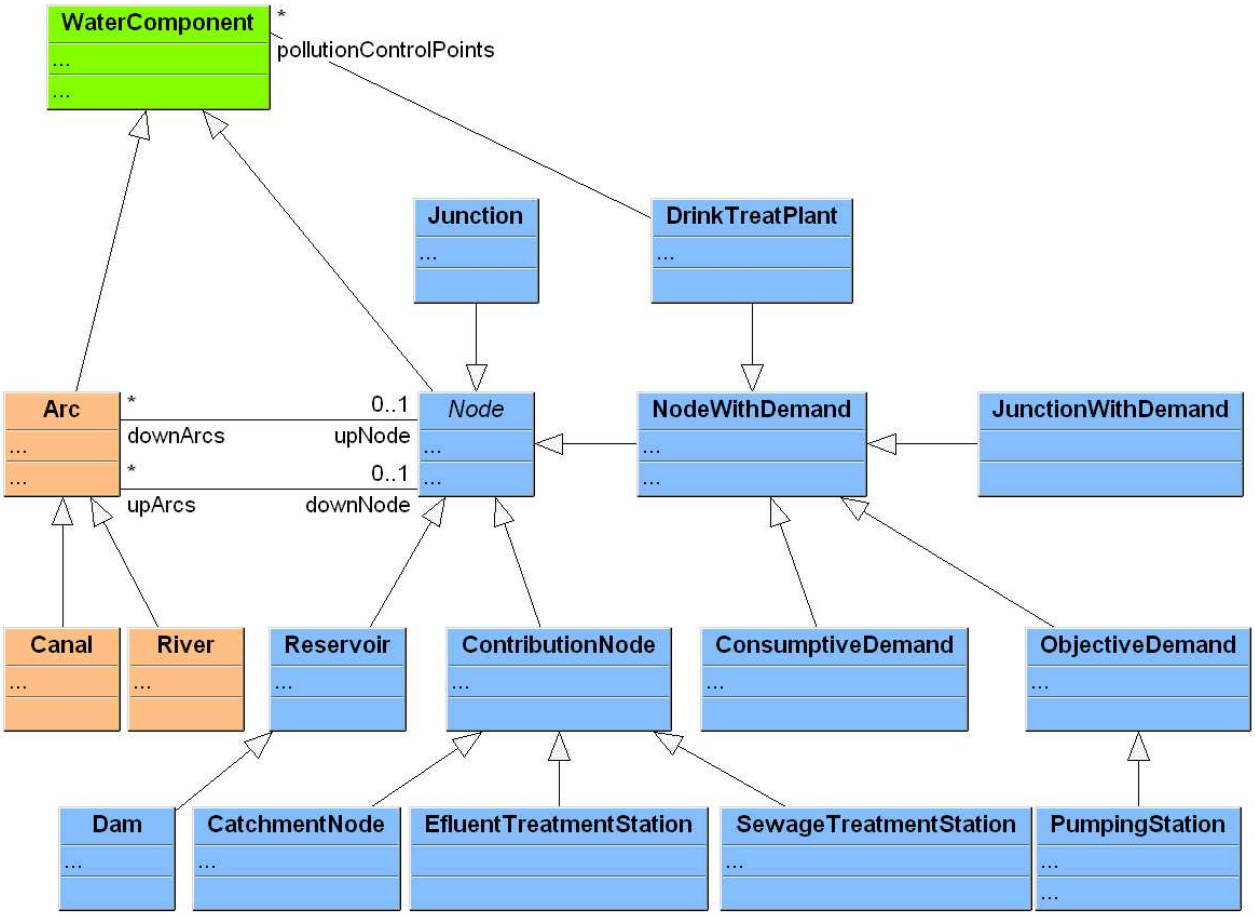


Figura 1: Visão geral do Agualoca

Na figura 1 pode-se ter uma visão geral da hierarquia das classes da parte hidrológica do Agualoca.

O diagrama UML completo do Agualoca pode ser visto dentro do próprio ambiente do VisualWorks[1] o quê permitiu a aplicação parcial do "*round trip engineering*".

WaterComponent é uma classe abstrata que herda da classe CORMAS ObjectLocation, uma classe usada no CORMAS para as assim chamadas entidades passivas. De WaterComponent especializamos as classes Arc e Node seguindo a a equipe hidrológica do projeto que utiliza essa forma de modelagem para uma bacia hidrográfica.

[1]Para isso utilizou-se a *parcel* ADvance2 provida pela IC&C GmbH, Software Foundations

3 Arcos

Esta seção descreve a modelagem e implementação da classe "Arcos" no programa Agualoca.

Embora não usada diretamente no programa, esta classe, mãe de "Canal" e "Rio" contém métodos específicos que por não serem redefinidos por suas classes filhas serão descritos nesta seção.

No Agualoca as diferenças entre "Canal" e "Rio" são aproveitadas basicamente para identificar o objeto "Rio" no caso de excesso de água nos vertedouros de represas.

3.1 Modelagem de Arcos — Objeto Arc

O objeto Arc foi modelado como uma subclasse de WaterComponent.

3.2 Implementação Smalltalk no Agualoca

3.2.1 Método step

No Agualoca os arcos recebem a informação da água em suas entradas através do método deliverWater dos nós a montante, pelo seletor inflow:.

No método step os arcos passam a água para as suas saídas na variável outflow fazendo o cálculo de perdas entre a entrada e saída do arco conforme a equação:

$$outflow = inflow(1 - lossCoef) \tag{1}$$

onde $1 \geq loosCoef \geq 0$ é um valor que caracteriza a perda de água ao longo do percurso do arco, seja devido a infiltração ou evaporação.

Outrossim, há necessidade de calcular se a vazão de água entrante é maior do que a capacidade do arco e proceder de acordo[2]. O excesso de água é considerado como um transbordamento do canal ou rio a ser simulado e subtraido da água disponível na entrada. A quantidade excedente é colocada na variável surplus que será depois disponibilizada para apresentação de resultados via método probe dos objetos.

Assim, o método step faz os testes e atribuições:

[2]Neste momento não temos claro pela documentação como o arco deve comportar-se quando $lossCoef > 0$ e $inflow > capacity$. Como solução de contorno, pode-se ver se o transbordamento dar-se-ia no meio do arco com o valor da perda calculado pela metade. Na presente implementação esse cálculo mais sofisticado *não é feito*.

if $inflow > capacity$ **then**
  $outflow \leftarrow capacity$
  $surplus \leftarrow inflow * (1 - lossCoef) - capacity$
  $deficit \leftarrow 0$
else
  $outflow \leftarrow inflow * (1 - lossCoef)$
  $surplus \leftarrow 0$
  $deficit \leftarrow capacity - inflow$
end if

De posse dessas variáveis atualizadas, este método chama então o método computeRetentionTime cuja implemenção é descrita na seção 3.3.1.

3.3 Cálculo das Concentrações de Poluentes

Os arcos no modelo Agualoca portam água entre os nós que trazem diversos tipos de poluentes. Um dos cálculos que o simulador deve fazer é a variação da concentração dos poluentes ao longo do comprimento dos arcos.

No caso do Agualoca assume-se que o fator limitante da eutrofização é o fósforo. Por essa razão usa-se esse nutriente como determinador da qualidade d'água.

A concentração de fósforo decai exponencialmente em função do tempo conforme:

$$c = c_o e^{-KT_r} \tag{2}$$

c: concentração final [kg/m^3];

c_o: concentração inicial [kg/m^3];

T_r: Tempo de retenção [mês] (A unidade de tempo igual a um *step* no Agualoca);

K: Coeficiente de decaimento do fósforo [mês^{-1}].

O tempo de retenção é dado por:

$$T_r = \frac{l}{2592U} \tag{3}$$

l: comprimento do arco [km];

U: velocidade média da água no arco [m/s].

A constante 2592 é necessária para que o resultado de T_r seja dado em meses com a velocidade e comprimento do arco dados com as unidades acima.

Como os arcos no modelo Agualoca modelam condutos abertos, a determinação da velocidade não é direta.

Fazendo a aproximação de a calha de corrimento de água dos arcos terem formato trapezoidal fig. 3.3, a velocidade pode ser determinada pela fórmula empírica proposta por Manning:

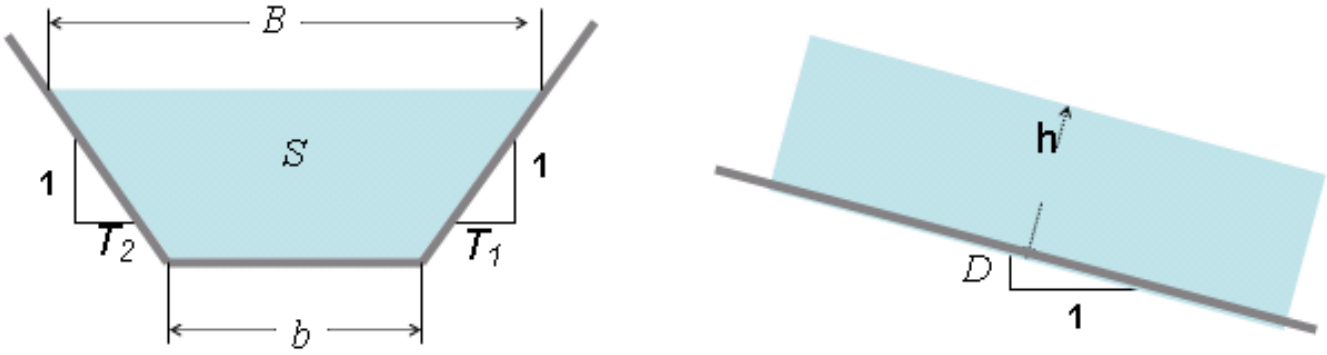


Figura 2: Idealização de uma calha do leito de um rio ou canal aberto. O valor de B, a base maior do trapézio formado pela seção de água corrente não é usado diretamente mas sim calculado a partir de b, T_1, e T_2 na equação 6.

$$U = \frac{1}{n} R^{2/3} \sqrt{D} \tag{4}$$

n: número de Manning (vide tabela 3 na pág. 25);

R: relação entre área da secção de água no arco e o perímetro molhado [m];

$$R = \frac{S}{P} \tag{5}$$

S: Secção da água no arco [m^2];

P: Perímetro molhado do curso d'água [m];

$$S = hb + h^2(T_1 + T_2) \tag{6}$$

onde:

h: altura da lâmina d'água [m];

b: base menor [m];

T_1: declividade do lado direito da calha [$^1/_1$];

T_2: declividade do lado esquerdo da calha [$^1/_1$].

$$P = b + h(\sqrt{1 + T_1} + \sqrt{1 + T_2}) \tag{7}$$

Temos ainda que levar em conta a vazão no conduto Q em m^3/s:

$$U = \frac{Q}{S} \tag{8}$$

Enquanto num conduto forçado a velocidade média pode ser calculada facilmente pela equação 8, em condutos abertos é necessário resolver de forma iterativa o sistema de equações formado pelas equações 8 e 4 acima para chegar a uma velocidade média para uma certa vazão no arco, pois a área (e a altura) da lâmina d'água são variáveis em função da velocidade (vazão).

O algoritmo para resolver a velocidade em função da vazão foi cedida ao projeto por integrantes do Laboratório de Suporte a Decisões – LabSid do Depto. de Enga. Hidráulica e Sanitária da Escola Politécnica de São Paulo.

A implementação em Visual Basic é mostrada na listagem do Apêndice A.

Trata-se de um algoritmo *Regula Falsi* com o uso de coeficientes de Fibonacci para acelerar a convergência do mesmo como descrito em [Mat]. Na seção 3.3.1 apresentamos a implementação no Agualoca.

3.3.1 Método computeRetentionTime

Este método é chamado a cada step do Agualoca para calcular a velocidade do rio ou canal em função do fluxo d'água e seu resultado serve para determinar o tempo de retenção conforme a equação 3. Como o valor de velocidade dos arcos não é empregado neste momento no projeto para nenhuma outra função que a determinação do tempo de retenção, esse valor não é armazenado ou reportado pelo Agualoca[3].

Na implementação do LabSid da rotina de cálculo da velocidade, cedida a este projeto, a criação da lista de números de Fibonacci é feita no início da rotina, com um tamanho de 50. Essa abordagem se aplicada ao Agualoca tem a seguinte implicação: como o método é chamado a cada step do Agualoca e para cada arco da simulação, temos para a rede hidrológica atual com 60 arcos e para uma simulação de dois anos um número de chamadas a esse código de $24 * 60 = 1440$ vezes.

Para evitar esse cálculo redundante, no Agualoca refatoramos esse código para um método chamado na inicialização do programa.

Outrossim, o único uso dos números de Fibonacci na rotina é para calcular as relações entre dois valores:

$\frac{Fib(N-Cont-2)}{Fib(N-Cont)}$

razão pela qual, na inicialização criamos diretamente uma lista com os valores das relações acima.

[3]Entretanto para fins de depuração seu valor pode ser lido nas mensagens do Transcript

4 Canais

4.1 Modelagem do Canal — Objeto **Canal**

O objeto Canal é uma subclasse de Arc

4.2 Implementação Smalltalk no **Agualoca**

4.2.1 Método **initCanals**

Este método é chamado na inicialização do Agualoca. A principal função deste método é carregar os parâmetros dos canais a partir de um arquivo, criando uma coleção de objetos do tipo Canal.

```
initCanals
  | auxCanal dataMatrix |
  dataMatrix := self getDataFromFile: 'canal.csv' separator: $,.
  dataMatrix do:
  [:ele |
  auxCanal := Canal new init.
  auxCanal name: (ele at: 2).
  auxCanal capacity: (ele at: 3) asNumber. "m/s"
  auxCanal slope: (ele at: 4) asNumber. "adimensional"
  auxCanal length: (ele at: 5) asNumber. "km"
  auxCanal lossCoef: (ele at: 6) asNumber. "adimensional"
  auxCanal upNode: (self findNode: (ele at: 7)).
  auxCanal downNode: (self findNode: (ele at: 8)).
  auxCanal coeffDecay: (ele at: 9) asNumber.
  auxCanal coeffManning: (ele at: 10) asNumber.
  auxCanal slopeT1: (ele at: 11) asNumber.
  auxCanal slopeT2: (ele at: 12) asNumber.
  auxCanal minorBase: (ele at: 13) asNumber.
  self theCanals add: auxCanal].
  self theCanals do: [:aCanal | aCanal addArcListToNodes]
```

5 Rios

5.1 Modelagem do Rio — Objeto River

5.2 Implementação Smalltalk no Agualoca

5.2.1 Método initRivers

Este método é chamado na inicialização do Agualoca. A principal função deste método é carregar os parâmetros dos rios a partir de um arquivo, criando uma coleção de objetos do tipo River.

```
  initRivers
| auxRiver dataMatrix |
dataMatrix := self  getDataFromFile: 'river.csv' separator: $,.
dataMatrix do:
[:ele |
auxRiver := River new init.
auxRiver name: (ele at: 2).
auxRiver capacity: (ele at: 3) asNumber. "m/s"
auxRiver slope: (ele at: 4) asNumber. "adimensional"
auxRiver length: (ele at: 5) asNumber. "km"
auxRiver lossCoef: (ele at: 6) asNumber. "adimensional"
auxRiver infiltrationLoss: (ele at: 7) asNumber. "adimensional"
auxRiver upNode: (self findNode: (ele at: 8)).
auxRiver downNode: (self findNode: (ele at: 9)).
auxRiver coeffDecay: (ele at: 10) asNumber.
auxRiver coeffManning: (ele at: 11) asNumber.
auxRiver slopeT1: (ele at: 12) asNumber.
auxRiver slopeT2: (ele at: 13) asNumber.
auxRiver minorBase: (ele at: 14) asNumber.
self theRivers add: auxRiver].
self theRivers do: [:aRiver | aRiver addArcListToNodes]
```

6 Nós

Esta seção descreve a modelagem e implementação da classe "Nó" nos Agualoca. Uma das classes fundamentais do modelo Agualoca da qual existem várias classes filhas especializando o comportamento para a modelagem dos objetos hidrológicos do simulador.

6.1 Modelagem de Nós — Objeto Node

O objeto Node é uma subclasse de WaterComponent.

6.2 Implementação Smalltalk no Agualoca

6.2.1 Método initNodes

Este método, chamado na inicialização, é chamado após todos os outros objetos do Agualoca tenham sido inicializados. Por essa razão esta inicialização é chamada por último, quando todas as informações dos nós e os arcos que os interligam já estão disponíveis.

A função deste método é criar uma coleção de objetos do tipo Node (incluindo aqui todos os objetos instanciados das classes filhas), na ordem hidrologicamente correta, para que no método step do Agualoca se possa iterar sobre essa coleção e chamar o método step de cada elemento dessa coleção e o programa produzir a simulação hidráulica.

A implementação chama o método postOrderNodeList descrito na seção 6.2.3.

Como opção, pode-se imprimir a lista ordenada dos nós, com a chamada ao método listToFile.

6.2.2 Método step

No modelo do Agualoca os nós são os responsáveis pela entrada, movimentação, armazenagem e saída de água no sistema (bacia).

Cada nó deve iterar a coleção de arcos a montante (upArcs) e acumular a água disponível nos outflow desses arcos na sua variável inflow.

Então, deve disponibilizar a água para os nós a jusante, seguindo as regras de distribuição conforme as prioridades e frações (v. método deliverWater) na seção 6.2.5.

Finalmente, cada arco a jusante deve ter seu método step chamado para que estes tenham, entre outras coisas, suas variáveis outflow atualizadas (para detalhes v. seção 3.2.1).

6.2.3 Método postOrderNodeList

A plataforma CORMAS constrói automaticamente para cada classe definida no modelo de usuário uma coleção (inicialmente vazia) cujo nome é feito começando com `the` e acrescentando um "`s`" ao final nome da classe. No

caso da classe Node temos então a coleção theNodes. Nessa coleção são colocados os nós ordenados conforme o algoritmo do método postOrderNodeList.

Para que o Agualoca faça a simulação da parte hidrológica do modelo ou "a água correr", é necessário que os nós sejam percorridos na ordem correta[4] para que o método step do programa ao simular o andamento dos meses faça as transferências dos volumes de água corretamente.

Em particular, os nós devem ser percorridos desde os nós de contribuição em direção aos nós de consumo. Um dado nó só deve ter seu mês, ou seja o step avançado (significando o andamento da simulação para êle) se todos os nós a montante já tiverem sido atualizados, desta forma tendo as vazões dos arcos com os valores de fluxo de saída resolvidos.

Este tipo ordenamento denomina-se "pós-ordem" (*post order*) e o algoritmo empregado no Agualoca apresentado no algoritmo 1.

Algoritmo 1 postOrder v

```
theNodes add v
for all w in upwardNodes v do
  if w not in theNodes then
    theNodes removeLast
    postOrder w
  end if
end for
for all u in downNodes v do
  postOrder u
end for
```

Algoritmo (*post order*) adaptado às necessidades do Agualoca. A lista *theNodes* na implementação é uma coleção do Agualoca. O uso dos métodos add e removeLast tornam desnecessário o uso de "labeling" como "colorir" as arestas visitadas nos algoritmos canônicos.

6.2.4 Método myDemands

Este método prepara uma estrutura de dados utilizada pelo método deliverWater (v. 6.2.5) que efetua a distribuição de água entre os arcos a jusante.

A distribuição da água é feita conforme certas regras que são executadas pelo algoritmo 3. Os atores atuam em variáveis via a IHM que em seu conjunto refletem as decisões destes em relação a essas regras.

Para esclarecer estas regras necessitamos primeiro das seguintes definições:

Demanda Quantidade de água que um elemento hidráulico do Agualoca solicita a cada step. A demanda pode ser satisfeita ou não, dependendo da existência de água suficiente ou não para cada caso. Existem as seguintes demandas:

Demanda Própria É a demanda de água que o próprio nó que está distribuindo a água precisa para atender as necessidades de sua especialidade. No Agualoca temos os reservatórios que para armazenar água disputam-na com as outras demandas a jusante.

Demanda de Arcos É a demanda de água devido à vazão mínima necessário nos arcos a jusante do nó. São de dois tipos:

[4] Conforme comunição entre os membros do projeto, repeitando a relação "*amont-aval*" formalizada por Jean-Christophe Pouget.

Ecológica Aquela dada pela vazão mínima em rios.

Técnica A dada pela mínima necessária em canais.

Demanda Objetiva É a demanda de água que um nó impõe no sistema para alterar a distribuição de água natural na bacia. Por exemplo por meio de estações de bombeamento de água, restrições (válvulas), etc.

Demanda Consuntiva É a demanda de água que um nó solicita no sistema para retirar água da bacia, como um grande consumidor (indústria), estação de tratamento de água, etc.

Prioridade É um número ordinal que indica a preferência que as demandas devem ser satisfeitas quando o sistema detecta que não há água suficiente para satisfazer totalmente a somatória das demandas. No Agualoca as prioridades são do número menor para o maior.

No caso de reservatórios que têm volume acima do seu objetivo, a demanda Demanda Própria pode, em alguns meses, ser negativa.

No método myDemands as demandas são todas calculadas como se fossem dos arcos a jusante do nó em questão, mesmo quando elas possam vir dos nós alimentados por esses arcos.

As prioridades usadas são as dos arcos e do seu nó a jusante de cada arco.

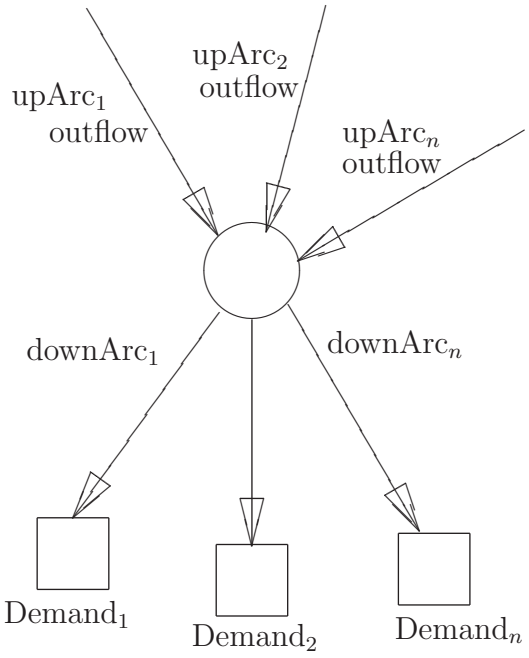


Figura 3: Objeto hidráulico Nó

Algoritmo 2 myDemands

```
for all downArcs do {seleciona os arcos que têm nós com demanda}
  if arc.downode é com demanda then
    selDems ← arc
  end if
end for
```

Algoritmo de construção da coleção myDemands.

6.2.5 Método deliverWater

O método deliverWater, chamado em cada step dos nós do Agualoca efetua a distribuição da água conforme as regras estabelecidas pelos atores do jogo, com as demandas coletadas em uma `Collection` conforme descrito em 6.2.4.

Uma vez que os nós têm suas demandas e prioridades estabelecidas (algumas destas dinamicamente durante o andamento da simulação), o algoritmo para executar a distribuição d'água usando das informações é o seguinte:

Algoritmo 3 deliverWater

$auxVol \leftarrow inflow$
for all downArcs **do** {zera entrada dos arcos}
  $Arc.inflow \leftarrow 0$
end for
for all priority in myDemands **do**
  if $auxVol \geq 0$ **then** {ainda há água}
    calcula $totalDemands$ {somatória das Demandas Totais da coleção}
    if $auxVol \geq totalDemands$ **then** {água de sobra}
      $redutor \leftarrow 1$
    else {água racionada}
      calcula $redutor = inflow/totalDemands$
    end if
    escala $totalDemands \leftarrow totalDemands * redutor$
    for all demands **do** {entrega Demanda multiplicada pelo redutor}
      $Arc.inflow \leftarrow demand * redutor$
    end for
    atualiza $auxVol \leftarrow auxVol - totalDemands$
  end if
end for
if $auxVol \geq 0$ **then** {sobrou ainda água}
  coloca no Rio a Jusante
end if

Algoritmo de distribuição de água do objeto Node. myDemands é uma coleção, indexada por prioridade, de coleções de demandas compostas por um objeto Arc e sua vazão demandada. Assim, para cada prioridade, havendo água, o algoritmo distribui água para essas demandas. No caso de a quantidade de água ser inferior à soma das demandas de uma determinada coleção em uma prioridade, `deliverWater` calcula o um redutor e fornece água de maneira proporcional a todas as demandas. Demandas menos prioritárias a partir desse ponto não recebem mais água. Por outro lado, se todas as demandas forem satisfeitas e ainda houver água, o restante será vertido no rio a jusante do nó.

A coleção ordenada *myDemands* na implementação é uma `OrderedCollection` (do Smalltalk) de coleções, contendo as demandas para cada prioridade e o ordenamento é feito pela prioridade.

Devido os nós, em geral, não terem capacidade de armanezamento, toda a água que sobre após o atendimento das demandas é vertido para o rio a jusante independente da capacidade dele.

7 Reservatórios

Esta seção descreve a modelagem e implementação da classe "Reservatórios" no Agualoca. Esta classe não usada diretamente no programa, sendo portanto uma classe virtual. No Agualoca ela é mãe de "Represa", no Spatmas ela também tem como subclasse "Várzea" que por não ser usada no Agualoca não é implementada neste modelo.

A descrição da modelagem hidrológica e as decisões sobre a representação deste objeto podem ser vistas na seção 4.2 de [Bet04].

7.1 Modelagem de Reservatórios — Objeto Reservoir

O objeto Reservoir foi modelado como uma subclasse de Node.

Do protocolo init ele tem seus dois métodos init e initd redefinidos pelas classes filhas.

7.2 Implementação Smalltalk no Agualoca

O objeto Reservoir acrescenta os atributos volume, maxMngVolume, minMngVolume, e volumeRec, e os respectivos métodos "gettter" e "settter".

volume Quantidade de água armazenada no Reservatório.

maxMngVolume Máximo volume administrável.

minMngVolume Mínimo volume administrável.

retentionTime Tempo de retenção da água no reservatório.

volumeRec Volume objetivo do gestor que maneja o reservatório;

8 Represas

Esta seção descreve a modelagem e implementação de represas no Agualoca. Na seção 8.1 percorremos a documentação já produzida ou recolhida pelo projeto, tanto para a modelagem para o programa como dados utilizados para decisões de modelagem. A seção 8.2 apresenta a modelagem do **objeto** Dam no programa Agualoca, e seu relacionamento com os outros objetos no modelo. No apêndice B.2 apresentamos a abordagem para o ajuste das curvas Cota versus Volume para as funções a serem usadas no modelo. A seção 8.6 apresenta os detalhes do modelo tanto na sua parte matemática como computacional implementados na plataforma CORMAS em Smalltalk.

8.1 Documentação Existente

A modelagem das represas — objeto Dam — no Agualoca é uma modificação da modelagem baseada nas decisões tomadas nas reuniões do projeto Negowat em Cochabamaba[Bet04], modelagem feita para os processos hidrológicos da Cabeceira do Alto Tietê[Cla03].

8.2 Modelagem da Represa — Objeto Dam

O objeto Dam foi modelado como uma subclasse de Reservoir.

Do protocolo WaterDynamic ele redefine os métodos deliverWater, getConcentrationP putDataOnNode e receiveP.

8.2.1 Equacionamento para Ajuste das Curvas Cota versus Volume

No Agualoca fizemos uso da modelagem decidida no Spatmas. A calibração é descrita no apêndice B.2

A análise numérica nos dados do Spatmas mostrou que a melhor aproximação (melhor correlação) para os dados das cinco represas disponíveis para a Bacia do Alto Tietê era o polinômio de quarto grau:

$$cota(v) = av^4 + bv^3 + cv^2 + dv + e \tag{9}$$

onde v é o volume da represa em milhares de metros cúbicos, *cota* é a altura em metros da lâmina d'água e a, b, c, d e d são coeficientes a ser ajustados para o conjunto de dados de cada represa, e que serão inicializados no método initDams do objeto Agualoca vide seção 8.6.

Essa equação foi verificada e validada com os dados disponíveis para o Spatmas.

8.3 Cálculo das Concentrações de Poluentes

De forma análoga ao objeto Arc descrito na seção 3.3, assume-se como fator limitante da eutrofização o fósforo, e esse nutriente como indicador de qualidade d'água.

No caso de reservatórios a concentração de fósforo decai exponencialmente em função do tempo conforme o modelo de Vollenweider:

$$P = \frac{L10}{V(\frac{1}{T_r} + K_s)} \tag{10}$$

P: concentração de fósforo no corpo d'água [gP/m^3]

L: carga afluente de fósforo [kgP/ano]

V: volume da represa [m^3]

T_r: tempo de retenção hidráulica [ano]

K_s: coeficiente de perda de fósforo por sedimentação [ano^{-1}]

Q_{out}: Vazão de saída da represa.

O tempo de retenção é dado por:

$$T_r = \frac{V}{12Q_{out}} \tag{11}$$

No caso do Agualoca há necessidade da divisão por 12 em função das unidades usadas internamente darem o tempo em meses.

8.4 Estratégias de Gestão

Em função do período do ano e necessidades de cada ponto específico na bacia, o gestor do reservatório pode selecionar uma *estratégia* que consiste em modificar as prioridades das vazões objetivas dos nós alimentados pelo reservatório. As estratégias disponíveis no Agualoca são inicializadas a partir do arquivo `damStrategy.csv` e estão apresentadas na tabela 1.

Tabela 1: Estratégias

Nome da estratégia	Vazão mínima à jusante	Vazão objetivo à jusante	Vazão mínima de transferência	Vazão objetivo de transferência	Estocagem
Esvaziamento	1	2	1	3	4
Estocagem	1	2	2	3	1
Abastecimento Seca	1	2	1	2	3
Abastecimento Cheia	3	3	1	1	2
Seca Extrema	1	4	1	4	4
Diluição efluente	1	2	2	3	4

As vazões objetivo nas colunas da tabela 1 se traduzem para os seguintes métodos dos objetos no código do Agualoca conforme a tabela 2.

Tabela 2: Vazões Objetivo

Vazão mínima à jusante	`River≫minFlow`
Vazão objetivo à jusante	`JunctionWithDemand≫objFlow`
Vazão mínima de transferência	`Canal≫minFlow`
Vazão objetivo de transferência	`ObjectiveDemand≫objFlow`
Estocagem	`Dam≫volumeRec - Dam≫volume`

8.5 Faixas de Volume Objetivo

Devido ao Agualoca ter uma "rodada" de seis meses (*step*), o volume objetivo precisa mudar a cada mês. Para isso, usa-se uma faixa dada por valores mínimos e máximos para cada mês calendário, como as curvas típicas mostradas na figura 4.

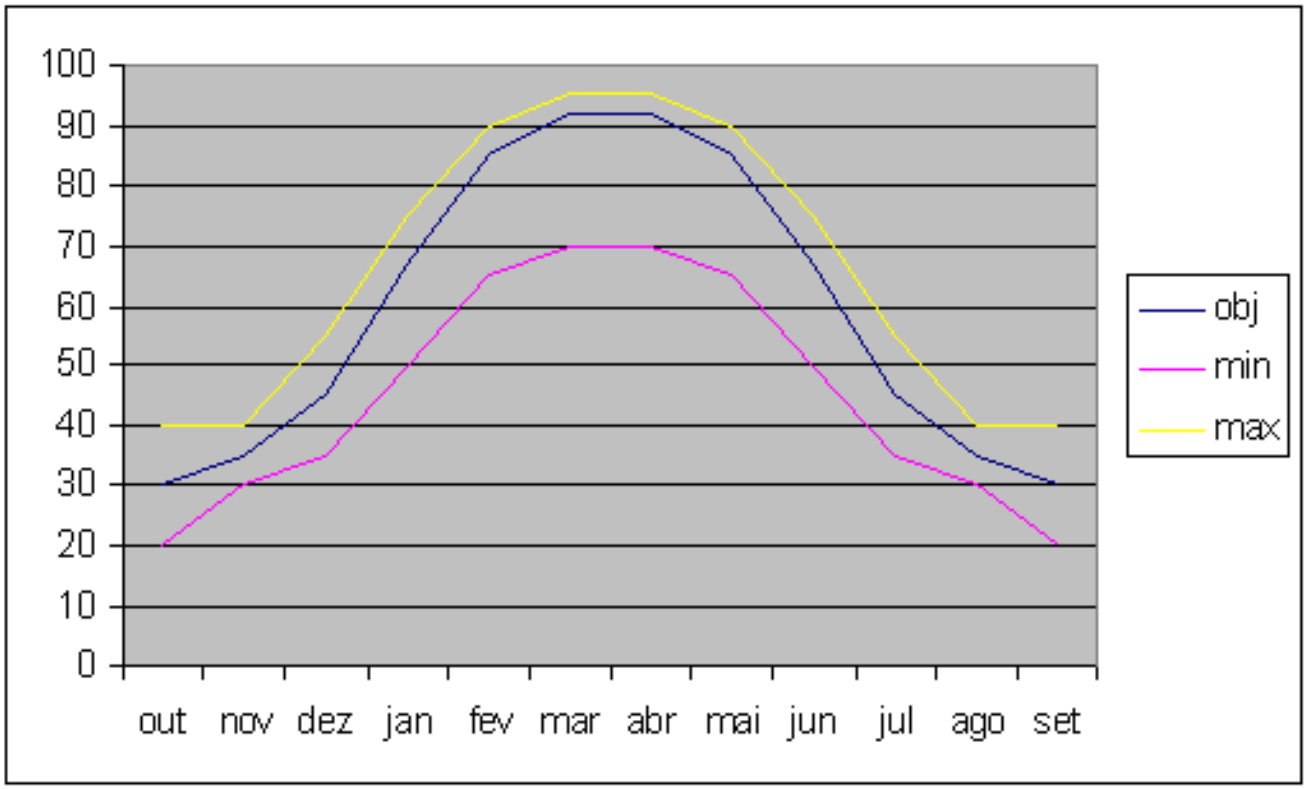


Figura 4: Curvas de mínimo e máximo volume objetivo por mês calendário para represas

O método deliverWater descrito na seção 8.6.2 emprega os valores dessas curvas.

8.6 Implementação Smalltalk no Agualoca

A modelagem das curvas de cota *versus* volume conforme mostrado na seção B.2.1 é utilizada no método damcota do objeto Dam.

8.6.1 Método retentionTime

Este método é chamado a cada step do Agualoca para calcular o tempo de retenção em função do fluxo d'água conforme a equação 11.

8.6.2 Método deliverWater

O médodo deliverWater para represas é modificado para que ele leve em conta a demanda própria desse tipo de nó e as estratégias de manejo do reservatório.

Em particular, ele se comporta como um controlador "*bang-bang*" tentando manter o volume (cota) dentro da banda indicada pelas curvas min e max na figura 4. Demandas cuja prioridade seja mais alta que a da própria necessidade de armazemento da represa são atendidas com a água existente na represa até o mínimo volume administrável minMngVolume. As demais demandas são atendidas enquanto o volume estiver acima do valor porcentual min conforme a figura 4 para o mês em questão, usando a repartição proporcional semelhante à do algoritmo 3.

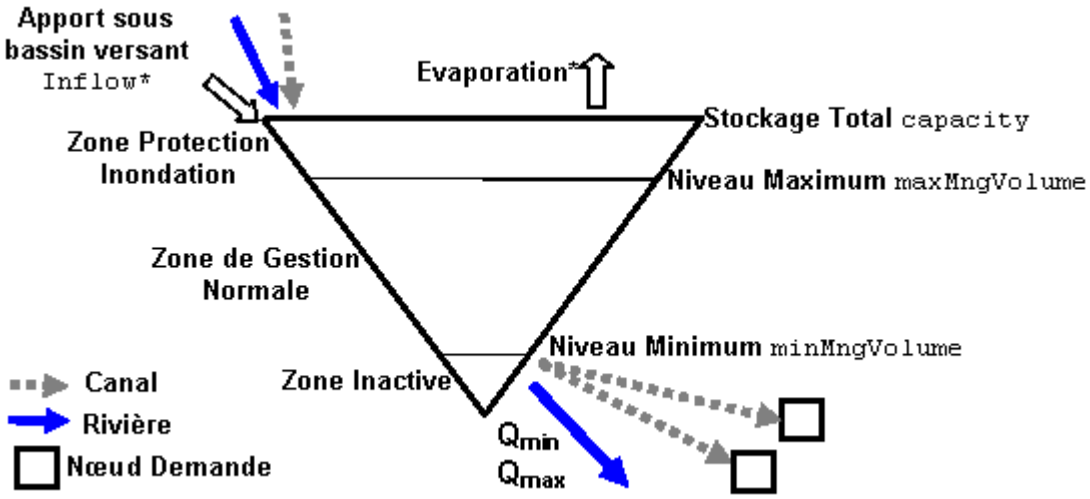


Figura 5: Objeto hidráulico Reservatório

Representação simplificada de um reservatório indicando os volumes e vazões que parametrizam este objeto no Agualoca. Os nomes em `courier` são os nomes dos atributos do objeto Smalltalk Reservoir no modelo. Adicionalmente, existe um atributo `volumeRec`, cujo valor deve estar entre `minMngVolume` e `minMngVolume`, que é o objetivo do volume do reservatório indicado pelo gestor, e o atributo `volume` que indica o estado de volume da represa no determinado `step` do modelo.
(*) No caso do Agualoca o valor de entrada já desconta a evaporação.

No caso de reservatórios, devido à sua capacidade de armazenamento, eventual água que sobre após o atendimento

Algoritmo 4 deliverWater

```
volume ← volume + inflow
for all downArcs do {zera entrada dos arcos}
  Arc.inflow ← 0
end for
for i ← 1 to myDemands.size do
  if i < priority then
    [prioridades maiores que a represa] auxVol ← volume − minMngVolume
  else
    if volume > volRecMin then {prioridades menores que a represa}
      auxVol ← volume − volRecMin
    else
      auxVol ← 0
    end if
  end if
  if auxVol > 0 then {há água p/distribuir}
    calcula totalDemands{somatória das Demandas Totais da coleção}
    if auxVol ≥ totalDemands then {água de sobra}
      redutor ← 1
    else {água racionada}
      calcula redutor ← inflow/totalDemands
    end if
    for all demands in myDemands[i] do {entrega Demanda multiplicada pelo redutor}
      Arc.inflow ← demand ∗ redutor
    end for
    atualiza volume ← volume − totalDemands ∗ redutor
  end if
end for
```

Algoritmo de distribuição de água do objeto Dam. myDemands é descrita na subseção 6.2.4. Demandas menos prioritárias que a demanda própria da represa somente recebem água se houver água acima do volume objetivo mínimo para o mês (*step*) considerado, dado pela variável *volRecMin*, calculado escalando-se a capacidade da represa pelo valor de min conforme a curva (específica para cada represa) mostrada na figura 4. As demandas mais prioritárias são atendidas até o limite de mínimo volume gerenciável da represa dado por minMngVolume na figura 5

de todas as demandas pode ser vertida ao rio a jusante ou armazenada até o limite `capacity` em função da vazão já calculada para o rio. Para tal, o método executa o algoritmo 5 tentando manter o rio fora do risco de enchentes e a cota entre `maxMngVolume` (v. fig. 5) e `capacity` na figura essa faixa é denominada de "zona de proteção à inundação", e somente se este volume estiver para ser ultrapassado é que o rio é inundado.

Algoritmo 5 spillWater *volume*

if $volume > volRecMax$ **then** {sobrou ainda água}
  calcula reserva de vazão dos arcos a jusante $totdem \leftarrow (myRiver.slack + myCanal.slack)$
  $aux \leftarrow min((volume - volRecMax), totdem)$
  $myRiver.inflow \leftarrow myRiver.slack * aux/totdem$
  $myCanal.inflow \leftarrow myCanal.slack * aux/totdem$
  $volume \leftarrow volume - aux$
end if
if $volume > capacity$ **then** {ainda acima capacidade}
  $myRiver.inflow \leftarrow myRiver.inflow + (volume - capacity)$
  $volume \leftarrow capacity$
end if

Algoritmo para verter a água restante. Primeiramente verifica-se a sobra (*slack*) de capacidade dos arcos (rio e canal) a jusante da represa. De forma análoga ao algoritmo 4, a variável $volRecMax$ é calculada escalando-se a capacidade da represa pelo valor de max da curva na figura 4. Tenta-se então distribuir a água que ultrapasse esse volume superior de forma proporcional entre eles. Se essa distribuição não trouxer o volume dentro da região de controle de enchentes, o adicional é jogado no rio, sem nenhuma crítica adicional, mesmo que cause o transbordamento dele.

8.6.3 Método initDams

Este método é chamado na inicialização do Agualoca. A principal função deste método é carregar os parâmetros das represas a partir de um arquivo, criando uma coleção de objetos do tipo Dam.

```
initDams
    | dataMatrix |
    self initDamStrategy.
    dataMatrix := self getDataFromFile: 'dam.csv' separator: $,.
    dataMatrix do:
        [:ele | | auxDam auxMatrix |
    auxMatrix := self getDataFromFile: (ele at: 2) , '.csv' separator: $,.
    auxDam := Dam new init.
    auxDam
      name: (ele at: 2);
      volume: (ele at: 3) asNumber / 2592; "m equivalentes mês"
      capacity: (ele at: 4) asNumber / 2592; "m equivalentes mês"
      minMngVolume: (ele at: 5) asNumber / 2592; "m equivalentes mês"
      maxMngVolume: (ele at: 6) asNumber / 2592; "m equivalentes mês"
      coeffC: (ele at: 9) asNumber;
      coeffD: (ele at: 10) asNumber;
      coeffE: (ele at: 11) asNumber;
      priority: (ele at: 12) asNumber;
      volumeRec: (ele at: 14) asNumber / 2592; "m equivalentes mês"
      fullname: (ele at: 15);
      chargeP: (ele at: 16) asNumber * auxDam volume * 1000; "conv concentra,cão em carga total"
      strategyDry: (ele at: 17);
      strategyRain: (ele at: 18);
      putDataOnNode: auxMatrix.
    auxDam initLogDAB: localLogProtocol.
    auxDam capacity > 0 ifTrue: [auxDam operation: 'Operacional'].
    self theDams add: auxDam]
```

A leitura do arquivo por getDataFromFile coloca as linhas da planilha de valores separados por vírgulas `dam.csv` na coleção ordenada dataMatrix, onde cada elemento dessa coleção é uma célula da planilha e o índice corresponde à sua coluna.

A segunda chamada a getDataFromFile obtém para cada represa a tabela de valores específicos para cada mês dos volumes objetivo mínimos e máximos (v. fig. 4).

A ordem de colocação dos dados na planilha de valores deve ser das represas de altitude mais alta para as de altitude mais baixa, para que a apresentação de seus dados na IHM do Agualoca seja correta (aproximadamente de montante a jusante) do ponto de vista hidrológico.

Assim para cada represa, os coeficientes da equação 9 são carregados a partir dos valores das colunas 9 – 11 da planilha.

Devido a uma decisão de projeto de ter os débitos de água indicados em metros cúbicos por segundo, e para não confundir os usuários na manutenção das planilhas, os valores dos volumes das colunas 3 – 6 e 14 são pré-escalados dividindo-se pela constante 2592 para converter milhares de metros cúbicos em um volume equivalente ao volume para a vazão dada em um segundo. Em outras palavras, o modelo trabalha com volumes de represa em escala para evitar proliferação de constantes em todos os cálculos.

Nos métodos da categoria probe do CORMAS implementados no Agualoca os volumes são re-escalados para que se veja a capacidade da represa real[5].

8.6.4 Método damcota:

Neste método se implementa o cálculo da equação 9. No Agualoca como se criou um conjunto de represas "sintéticas", ficou decidido que as curvas de ajuste são do segundo grau.

A colocação os coeficientes multiplicados pelas potências de aVolume vem da sintaxe do Smalltalk para evitar uma quantidade muito grande de parênteses.

```
damcota: aVolume
    ^aVolume ** 2 * self coeffC + (aVolume * self coeffD) + self coeffE
```

Os coeficientes empregados na inicialização das represas são os corretos para calcular as cotas com o volume escalado como mencionado na subseção anterior.

8.6.5 Método cotaToVol:

Este método é usado pela IHM para estabelecer cotas objetivo e outras mudanças no regime hidrológico.

Calcula o volume correspondente a uma cota dada usando o método numérico *Regula Falsi*.

Permite assim que se obtenha a função inversa à dada pela equação 9 sem ter que obter e calibrar mais um conjunto de coeficientes.

Para evitar que o programa se perca com cotas fora da faixa de cada represa uma crítica e coerção a valores mínimo e máximo é feita no início.

[5] Essa técnica de apresentar dados nas diversas formas mantendo uma única representação interna é denominado *Princípio do Acesso Uniforme* por Bertrand Meyer [Mey97] e é incentivado pela tecnologia Smalltalk

```
cotaToVol: aCota
    | accuracy aVol1 aVol2 dif aVol xCota lCota hCota |
    lCota := self damcota: 0.
    hCota := self damcota: capacity.
    (aCota between: lCota and: hCota)
        ifTrue:
            [accuracy := 1.0e-4.
            aVol1 := 0.
            aVol2 := capacity.
            dif := 1.
            [dif abs > accuracy] whileTrue:
                [aVol := (aVol2 + aVol1) / 2.
                xCota := self damcota: aVol.
                dif := xCota - aCota.
                dif > 0 ifTrue: [aVol2 := aVol] ifFalse: [aVol1 := aVol]]]
        ifFalse:
            [aCota < lCota ifTrue: [^0].
             aCota > hCota ifTrue: [^capacity]].
    ^aVol
```

A Listagem Aquanet

```
Function Fibonacci(ByVal Q As Double,
                   ByVal nMan As Double,
                   ByVal Decliv As Double,
                   ByVal BaseMenor As Double,
                   ByVal T1 As Double,
                   ByVal T2 As Double) As Double
    Dim A As Double
    Dim Area As Double
    Dim B As Double
    Dim Delta As Double
    Dim Dif As Double
    Dim l As Double
    Dim Per As Double
    Dim Precisao As Double
    Dim Q1 As Double
    Dim Q2 As Double
    Dim X1 As Double
    Dim X2 As Double
    Dim F(50) As Double
    Dim Cont As Integer
    Dim i As Integer
    Dim N As Integer
    N = 50
    A = 0
    B = 100
' Implementação da recursão de Fibonacci
    F(0) = 1
    F(1) = 1
    For i = 2 To N
        F(i) = F(i - 1) + F(i - 2)
    Next i

    Precisao = 0.0000000001
    Delta = B - A
    Cont = 0
    Do
        l = F(N - Cont - 2) / F(N - Cont) * Delta
        X1 = A + l
        X2 = B - l
        Area = BaseMenor * X1 + T1 * X1 / 2 + T2 * X1 / 2
        Per = BaseMenor + X1 * ((1 + T1 ^ 2) ^ 0.5) + X1 * ((1 + T2 ^ 2) ^ 0.5)
        Q1 = (Area / nMan) * ((Area / Per) ^ (2 / 3)) * (Decliv ^ 0.5) - Q
        If Q1 < 0 Then
            Q1 = 0
        End If
        Area = BaseMenor * X2 + T1 * X2 / 2 + T2 * X2 / 2
        Per = BaseMenor + X2 * (1 + T1 ^ 2) ^ 0.5 + X2 * (1 + T2 ^ 2) ^ 0.5
        Q2 = Area / nMan * (Area / Per) ^ (2 / 3) *
             Decliv ^ 0.5 - Q
        If Q2 < 0 Then
            Q2 = 0
        End If
        Dif = Q1 - Q2
        If Q2 > Q1 Then
```

```
            B = X2
        Else
            A = X1
        End If
        Delta = X2 - X1
        Cont = Cont + 1
    Loop While Abs(Dif) > Precisao And N - Cont - 2 >= 0
    Fibonacci = X1
End Function
```

B Calibração

Como o Agualoca é a simulação de uma bacia hipotética, criada sinteticamente para atender os objetivos do jogo de papéis, certas premissas foram observadas para que o modelo tenha efetivamente similaridade com a realidade física (a água deve correr de cotas mais altas para mais baixas, exceto se houver bombeamento, por exemplo) e os elementos hidrológicos colocados possam ter realísticamente os atributos necessários para que a resposta do simulador com respeito aos fluxos de água, estoques nos reservatórios e qualidade sirvam para os objetivos do jogo.

Na subseção B.1 apresentamos as premissas e as decisões para os rios e canais do Agualoca. Na subseção B.2 o método usado para chegar à equação 9 é apresentado com os exemplos de dados do Spatmas vindos das represas da Bacia do Alto Tietê. Na subseção B.3 as escolhas dos regimes pluviométricos dos CatchmentNode.

B.1 Arcos

O ambiente representado pelo Agualoca tem um relevo estabelecido e uma bacia hidrográfica imaginada para proporcionar os regimes hidrológicos que quando manejados pelos atores do jogo criarão as situações que interessam aos objetivos do jogo.

Para manter verossimilhança do modelo as seguintes premissas são adotadas.

B.1.1 Comprimentos

1. Os comprimentos têm que ter coerência com a sua representação no mapa. A distãncia deles deve estar dentro da distância geográfica esperada para um arco conectanto as células de partida e de chegada. Valendo para os rios naturais e canais de transferência.
2. Para os canais de efluentes, os que conectam os "catchment nodes" e o canal que leva ao dreno, seus comprimentos serão arbitrariamente definidos como bem pequenos (0,01km).

B.1.2 Declividades

Rios As declividades são indispensáveis para estimar a velocidade seguundo Manning. Com referência à bacia do ATC, um layout de "altitudes" foi criado. As declividades encontradas na bacia foram estimadas a partir desse layout[6], e dos comprimentos estimados antes.

[6]Seria interesante colocar um atributo *altitude* nas células, e uma localização nos nós. Assim, como um arco está ligado a dois nós, ele pode calcular sozinho a sua declividade. Daria mais de elasticidade ao modelo.

Canais

1. Um valor por "default" de 0,001 foi colocado nos arcos ligados aos catchment nodes, arcos de efluentes, o dreno e os arcos representando a aspiração dos bombeamentos. Esses arcos servem de artifícios para inserir poluição nos rios. Esse valor influirá a velocidade, que influirá no tempo de retenção, que influirá no decaimento da poluição no arco. Para parar esse decaimento, o coeficiente de decaimento desses elementos será colocado em zero.
2. Os arcos de transferência são parecidos como rios, o cálculo foi efetuado da mesma forma que para os rios naturais.

Coeficientes de Manning Os valores encontrados na literatura e utilizados no Agualoca são os seguintes:

Tabela 3: Coeficientes de Manning

Material	nº Manning	Material	nº Manning
Rios		Canais	
Limpos e Retos	0,030	Leito Limpo	0,022
Rios Grandes	0,035	Leito Pedregoso	0,025
Lentos com Poços Profundos	0,040	Leito com ervas	0,035

Lista parcial dos coeficientes de Manning compilados por LMNO Engineering. O extrato aqui inclui tão somente as condições que interessam ao modelo Agualoca. A tabela deles apresenta valores para condutos metálicos, plásticos e outros materiais usados na Engenharia Hidráulica.

B.2 Represas

B.2.1 Ajuste das Curvas

As funções matemáticas expressando a relação entre as cotas e o volume nas represas nos foi passado na planilha [D.A05]. Essa relação, útil para os operadores das represas, não atende às necessidades da modelagem no Spatmas pois ele calcula os regimes de vazão nos rios e canais que alimentam e consomem água das represas obtendo assim os volumes nas represas e não suas cotas.

A obtenção das funções inversas, ou seja cota em função do volume, não pode ser feito de forma fechada, analiticamente, devido às funções enviadas pelo D.A.E.E. já serem resultado de ajustes de curvas válidas apenas para os intervalos de interesse de cada represa.

Assim, optou-se por utilizar as informações da planilha [D.A05] como fonte de *dados* gerando um conjunto de pontos razoável para cada represa e fazendo então um novo ajuste de curva para cada uma empregando uma função escolhida para padronizar a implementação no modelo computacional.

Como utilizamos o ajuste das curvas das represas no método damcota (v. 8.6.4 para calcular os valores das cotas conforme a simulação andar, precisamos empregar uma mesma equação geral para todos os ajustes e fazer a regressão para determinar os coeficientes particulares de cada represa[7].

[7] A alternativa seria criar mais um atributo no objeto Dam para entrar cada uma das equações o quê complicaria demasiadamente o modelo computacional do Agualoca.

A equação escolhida foi a 9 para a qual buscou-se uma correlação boa $R^2 \geq 0,9999$ pelos seguintes motivos:

- devido ao tamanho (volume) das represas um cm de cota representa boa quantidade de água[8]
- os dados vindos da planilha [D.A05] já são aproximações feitas para as relações cota x volume das represas e correlações com R^2 indicando correlações menores conduziriam a "alisamentos" nas curvas com conseqüentes erros em partes específicas do domínio da função.

Com a equação geral escolhida, gerou-se para cada represa uma folha de cálculo numa planilha auxiliar, cada uma com pontos a cada 5% do volume máximo das represas a partir dos dados do D.A.E.E.. O volume máximo de cada represa foi também obtido dos dados das planilhas que continham uma expressão para indicar a capacidade do reservatório de forma percentual. Esses dados foram então analisados e feita a regressão não linear usando-se ferramentas computacionais.

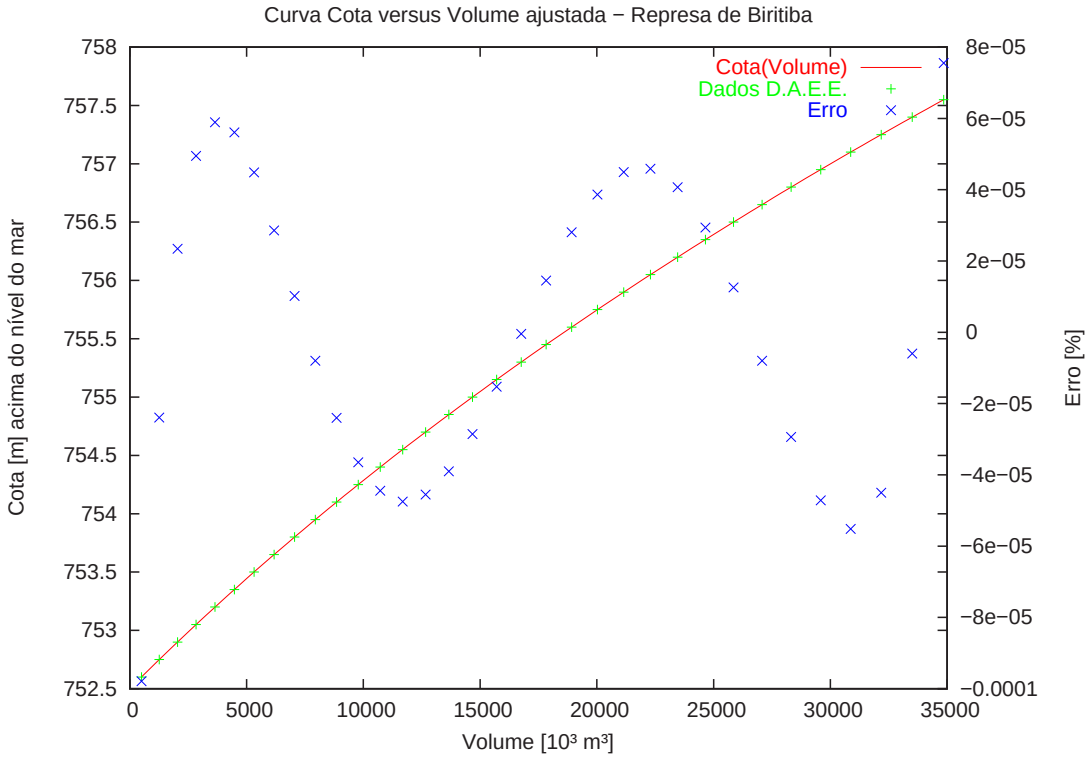


Figura 6: Biritiba

B.2.2 Resumo dos Resultados Obtidos

Na tabela 4 apresentamos resumidamente os resultados obtidos. O erro percentual calculado foi no pior caso 0,01% para a represa de Paraitinga e a melhor correlação e menor erro obtido para a represa de Biritiba. Não obstante o erro para Paraitinga ser cem vezes o erro obtido para o melhor caso ele é ainda assim, em valores absolutos, bem pequeno e mesmo considerando os volumes envolvidos não introduz erro adicional ao advindo dos próprios dados de campo[9]

[8] Em represas usadas para a geração de energia elétrica, os operadores das usinas pensam em termos de milímetros de lâmina d'água.

[9] É de se esperar que as cotas do reservatório não tenham uma precisão maior que ±1 cm. Outrossim, os dados não são informados com nenhuma correção devido a fatores de influência como temperatura, etc.

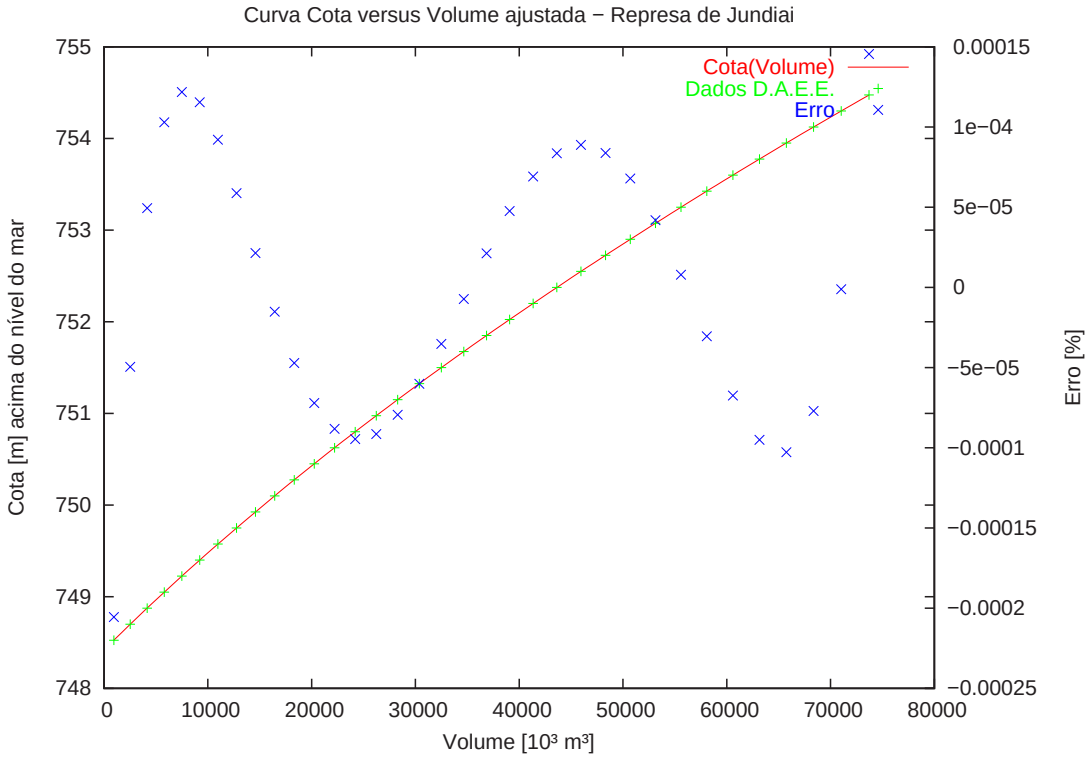


Figura 7: Jundiai

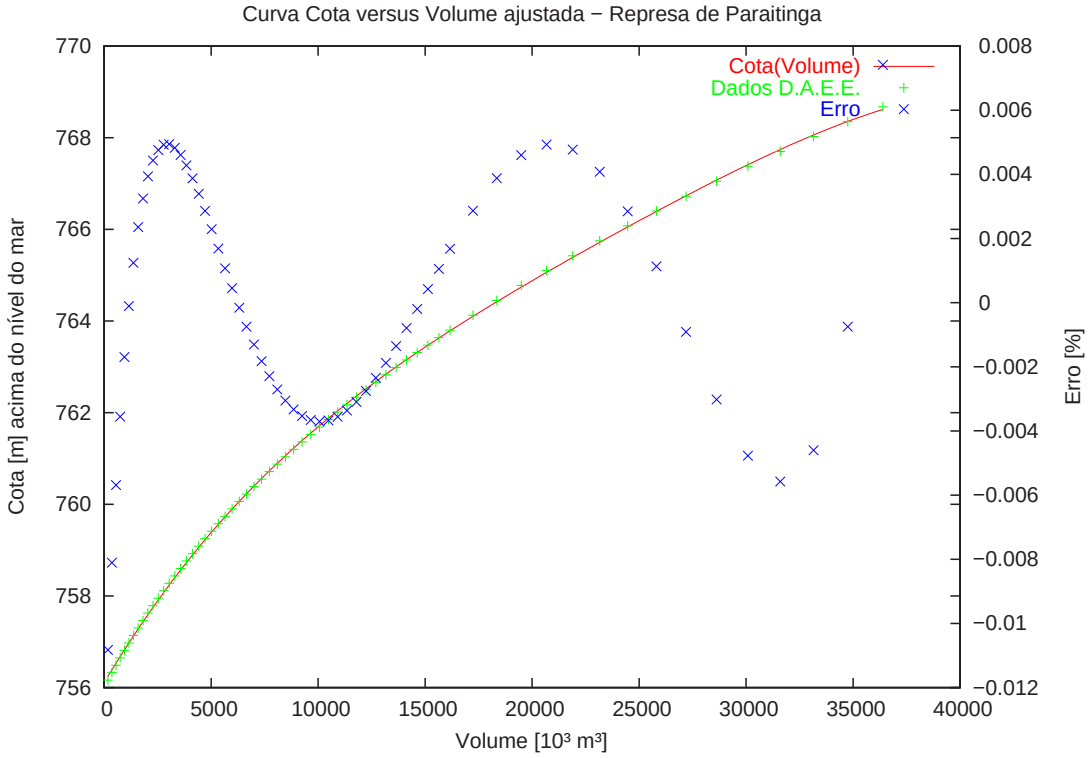


Figura 8: Paraitinga

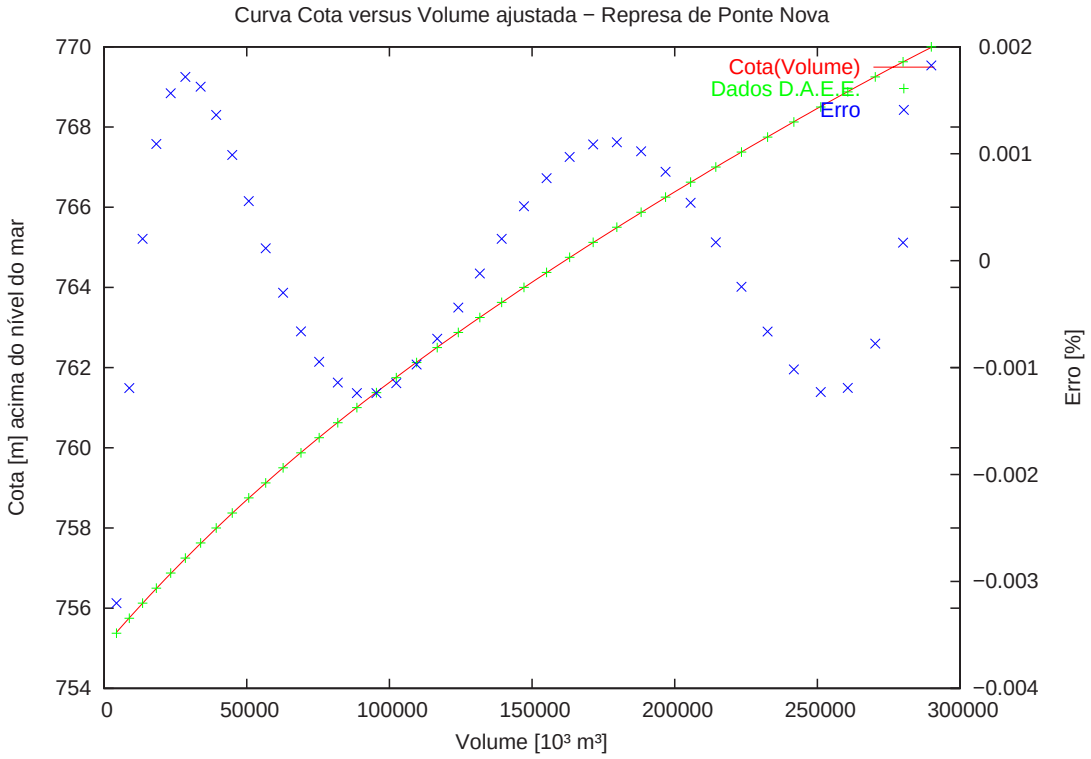


Figura 9: PonteNova

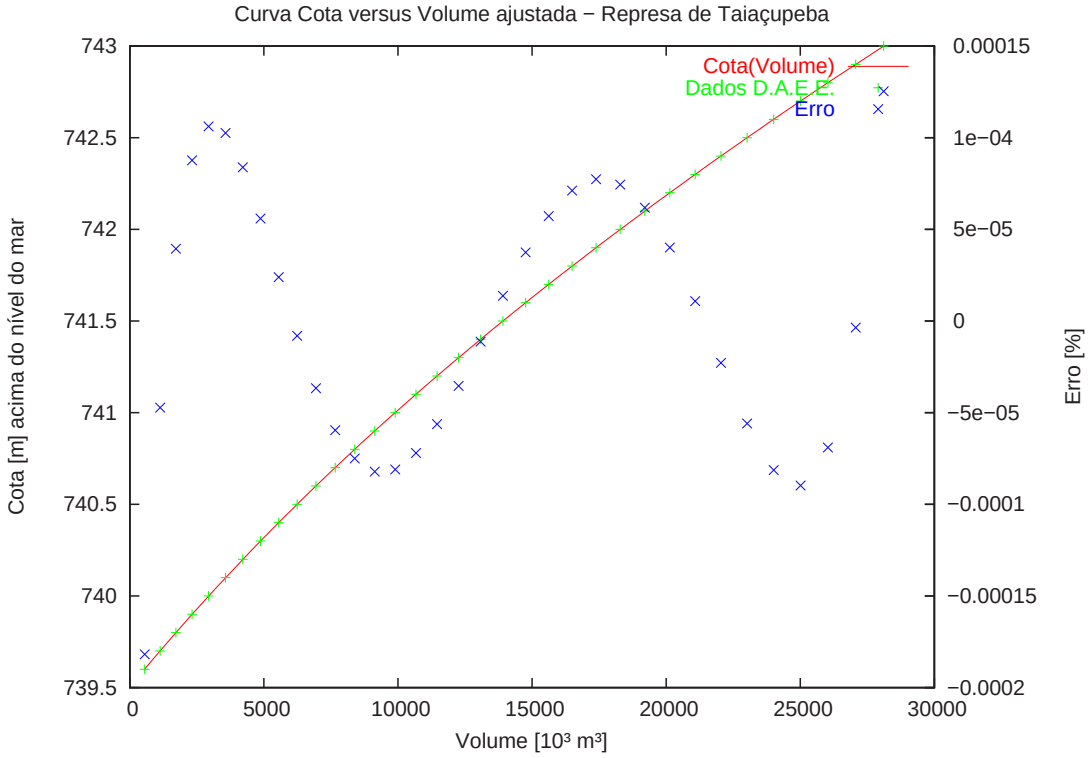


Figura 10: Taiaçupeba

Tabela 4: Resultados do Ajuste de Curvas das Represas

Represa	R^2	Erro máx.%
Biritiba	1,0000	0,0001
Jundiaí	0,9999	0,0002
Paraitinga	0,9999	0,0100
Ponte Nova	0,9999	0,0030
Taiaçupeba	0,9999	0,0002

Correlações e erros máximos calculados para o ajuste das curvas na equação geral 9. Os erros foram calculados fazendo a diferença entre o valor retornado pela equação ajustada e valor inicial da cota como dado vindo da planilha [D.A05].

B.3 Catchment Node

Escolha das séries de chuvas

B.3.1 Bacia 1

A bacia 1 tem uma superfície de 190 km^2, e fica bem a montante do sistema. Foi assumido que era parecida com a bacia Paraitinga, que está localizada a montante da bacia e tem uma área de drenagem de 186 km^2, os dados do arquivo Paraitinga.csv foram importados tal qual no arquivo CN1.csv.

B.3.2 Bacia 2

A bacia 2 tem uma superfície de 320 km^2. Ela apresenta as Alturas as maiores da bacia, o que é sinônimo de chuvas fortes. Nos arquivos do Spatmas, a soma das áreas das bacias PonteNova, RioClaroMontante, RioClaroJusante e RibeiraodoCampo dão um resultado de 306 km^2. Foi assumido que as contribuições da bacia 2 eram a soma dessas quatro bacias. O arquivo CN2.csv é a soma dos arquivos : PonteNova.csv + RioClaroJ.csv + RioClaroM.csv + RibdoCampo.csv.

B.3.3 Bacia 3

A bacia 3 tem uma superfície de 75 km^2, representa o primeiro trecho do rio principal (o trecho mais a montante do rio principal). A bacia TietêMontante do Spatmas aparece como similar a essa bacia: fica na parte montante do rio Tietê (o rio principal) e tem uma área de drenagem de 81 km^2. Os dados de contribuição do arquivo TieteM.csv foram importadas tal qual no arquivo CN3.csv.

B.3.4 Bacia 4

A bacia 4 tem uma área de drenagem de 120 km^2. A situação geográfica dela aparece como aquela da bacia Biritiba que tem uma área de drenagem de 78 km^2. Os dados do arquivo Biritiba.csv foram multiplicados por 1,5 para criar o arquivo CN4.csv.

B.3.5 Bacia 5

A bacia 5 tem uma superfície de 105 km^2 e representa o segundo trecho do rio principal, ou trecho meio. Os dados escolhidos para o arquivo CN5.csv foram os do arquivo TieteMogi.csv, por causa da localização do trecho de Mogi na bacia do ATC, meio do rio Tietê. Esses dados foram multiplicados por 1,2 porque a superfície da bacia TieteMogi tem 87 km^2 quando a bacia que participa a contribuição do CN4 tem 105 km^2.

B.3.6 Bacia 6

A bacia 6 tem uma área de drenagem de 120 km^2. A situação geográfica dela aparece como aquela da bacia Jundiai que tem uma área de drenagem de 109 km^2. Os dados do arquivo Jundiai.csv foram multiplicados por 0,9 para criar o arquivo CN6.csv.

B.3.7 Bacia 7

A bacia 7 é geograficamente localizada como a bacia 5. Ela tem uma área de drenagem de 175 km^2 e representa o terceiro trecho do rio principal. Como para a bacia 5, os dados escolhidos para encher o arquivo CN7.csv foram os dados do arquivo TieteMogi.csv, por coisa da localização do trecho de Mogi na bacia do ATC, meio do rio Tietê. Esses dados foram multiplicados por 2 porque a superfície da bacia TieteMogi tem 87 km^2 enquanto a contribuição da bacia CN7 tem 175 km^2.

B.3.8 Bacia 8

De situação geográfica, a bacia 8 pode ser comparada à bacia Taiaçupeba. A sua área de drenagem é de 150 km^2 quando a área de drenagem de do ponto Taiaçupeba é de 208 km^2. Para criar o arquivo CN8.csv, foram importados os dados do arquivo Taiaçupeba.csv multiplicados por 0,7.

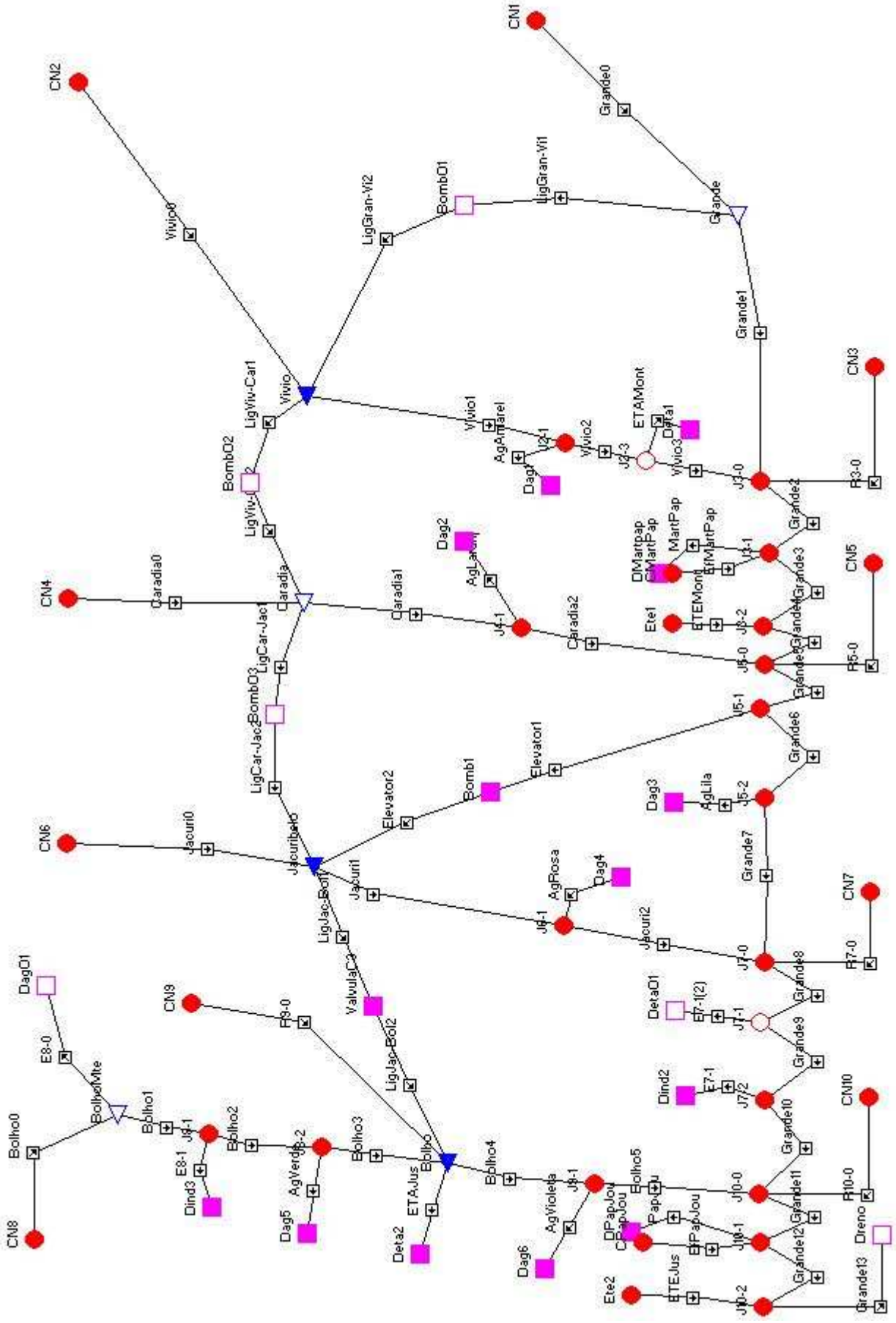
B.3.9 Bacia 9

As mesmas hipóteses foram feitas para a bacia 8 e a bacia 9. O arquivo CN9.csv é o mesmo do que o arquivo CN8.csv.

B.3.10 Bacia 10

A bacia 10 esta localizada a jusante da bacia fictícia. No modelo Spatmas, seria a bacia TieteMogi que mais corresponderia a essa localização. As contribuições do arquivo TieteMogi.csv foram importados multiplicadas por um fator de 0,75 porque a bacia TieteMogi tem uma superfície de 195 km^2 enquanto a bacia 10 tem uma superfície de 150 km^2.

C Rede Hidrológica



Referências

[BBPP98] F. Bousquet, I. Bakam, H. Proton, and C. Le Page. Cormas: Common-pool resources and multi-agent systems. *Lecture Notes in Artificial Inteligence*, 1416:826–838, 1998.

[Bet04] Bernardo Paz Betancourt. Réunion de modélisation hydrologique. Technical report, Negowat, fevereiro 2004.

[CIR02] CIRAD. *CORMAS Tutorial 2*, dezembro 2002. Building a Cormas model from scratch step by step.

[CIR03] CIRAD. *CORMAS Tutorial 1*, janeiro 2003. Manipulation of an existing example and creation of a new model.

[Cla03] Luci Clavel. Formalisation des processus hydrologiques en zone périurbaine; application au sous bassin de l'Alto Tietê Cabeceiras, São Paulo. Technical report, CIRAD, Montpellier, France, julho 2003.

[D.A05] D.A.E.E. 6 -monitoramento - junho 2005.xls. Planilha Excel, junho 2005.

[DPPT04] Raphaèle Ducrot, Bernado Paz, Jean-Christophe Pouget, and José Galiziano Tundisi. Le développemnet d'outils de simulation pour faciliter les concertations dans la gestion de bassins versants peri-urbains: exemple de São Paulo (Brésil). Discussão sobre simulador e jogo de papéis, setembro 2004.

[Duc04] Raphaèle Ducrot. Network of SPATMAS. Descrição da rede hidrológica e simbologia, maio 2004.

[Mat] John H. Mathews. Module for module for the Fibonacci search. WWW.

[Mey97] Bertrand Meyer. *Object-oriented software construction.* Prentice Hall, second edition edition, 1997.

[Rob02] Alexandre Nunes Roberto. Modelos de rede de fluxo para alocação da água entre múltiplos usos em uma bacia hidrográfica. Master's thesis, Escola Politécnica da Universidade de São Paulo, 2002.